Lecture 8
Introduction to point-based rendering

Dr. Lin ZHANG
School of Software Engineering
Tongji University
Spring 2012
Outline

• Motivation
• Point-based rendering
• Point-based surface editing
Motivation

• Why points? [1]
  • Point primitives have experienced a major “renaissance” in Graphics
  • Two reasons for that:
    • Dramatic increase in polygonal complexity
    • Upcoming 3D digital photography
  • Researchers start to question the utility of polygons as “the one and only” fundamental graphics primitive
  • Points complement triangles!

Motivation

• Triangle meshes based modeling
  • Pros
    • Simple and efficient representation
    • Hardware pipelines support
    • Advanced geometric processing
    • The widely accepted queen of graphics primitives
  • Cons
    • Sophisticated modeling is difficult
    • Complex LOD management
    • Compression and streaming is highly non-trivial
Motivation

• Triangles to points
  • Piecewise linear functions to Delta distributions
  • Discrete samples of geometry
  • No connectivity or topology – most simple
  • Store all attributes per surface sample
Motivation

• Point-based techniques are more suitable for complex scanned objects
  • Capturing geometry of such objects are not trivial
  • And sometimes representing the true geometry would be very inefficient (e.g. for hair). We would simply need many points to represent it
Motivation

• Modern 3D scanning devices (e.g., laser range scanners) acquire huge point clouds
• Generating consistent triangle meshes is time consuming and difficult
• A rendering primitive for direct visualization of point clouds, without the need to generate triangle meshes?
Motivation

- Modern 3D scanning devices (e.g., laser range scanners) acquire huge point clouds

David statue, about 4 million points
Digitized Michelangelo Project, Levoy et al., Stanford, 2000
Outline

• Motivation
• Point-based rendering
• Point-based surface editing
Point-based surface representation

- Point clouds instead of triangle meshes and textures as a rendering primitive [Levoy and Whitted 1985]
- Points in 3D are analogous to pixels in 2D

triangle mesh with textures  \rightarrow  point cloud, i.e., nonuniform surface samples
Point-based surface representation

- Points are non-uniform samples of the surface
- The point cloud describes:
  - 3D geometry of the surface
  - Surface reflectance properties (e.g., diffuse color, etc.)
- Points discretize geometry and appearance at the same rate
- There is no additional information, such as
  - connectivity (i.e., explicit neighborhood information between points)
  - texture maps, bump maps, etc.
Surface elements—surfels

- Each point corresponds to a surface element, or surfel, describing the surface in a small neighborhood
- Basic surfels

```cpp
BasicSurfel
{
    position;
    color;
}
```
Surfels

• Surfels can be extended by storing additional attributes

• This allows for higher quality rendering or advanced shading effects

\[
\text{ExtendedSurfel} \\
\{ \\
\text{position}; \\
\text{color}; \\
\text{normal}; \\
\text{radius}; \\
\text{etc...} \\
\}
\]
Surfels

• Surfels store essential information for rendering
• Surfels are primarily designed as a point rendering primitive
• They do not provide a mathematically smooth surface definition
General Point-based Rendering Framework

- Rendering pipeline
  - Simple, pure forward mapping pipeline
  - Surfels carry all information through the pipeline ("surfel stream")
  - No texture look-ups
General Point-based Rendering Framework

- Rendering pipeline

Input surface samples → Shade samples → Warp to image space → Continuous Reconstruction → Filtering, sampling

- "vertex shading"
- "vertex projection"
- "triangle rasterization"
- "texture filtering"
QSplat: a point-based rendering system

• Proposed in [1]

QSplat: a point-based rendering system

- An interactive viewer for large models ($10^8 - 10^9$ samples)
- Fast startup and progressive loading
- Maintains interactive frame rate
- Compact data structure
- Fast preprocessing
QSplat: a point-based rendering system

• QSplat data structure
  • Key observation: a single bounding sphere hierarchy can be used for
    • Hierarchical frustum and backface culling
    • Level of detail control
    • Splat rendering
QSplat: a point-based rendering system

• Creating the data structure
  • Start with a triangle mesh
QSplat: a point-based rendering system

- Creating the data structure
  - Place a sphere at each node, large enough to touch neighbor spheres
QSplat: a point-based rendering system

- Creating the data structure
  - Build up hierarchy
QSplat: a point-based rendering system

- Creating the data structure
  - Position and radius encoded relative to parent node
    - Hierarchical coding vs. delta coding along a path for vertex positions
QSplat: a point-based rendering system

• Creating the data structure
  • Normal quantized to grid on faces of a cube
  • Per-vertex color is quantized 5-6-5 (R-G-B)

52×52×6
QSplat: a point-based rendering system

- Traverse hierarchy recursively

```plaintext
if (node not visible)
    Skip this branch
else if (leaf node)
    Draw a splat
else if (size on screen < threshold)
    Draw a splat
else
    Traverse children
```
QSplat: a point-based rendering system

• Feedback-driven frame rate control
  • During motion: adjust recursion threshold based on time to render previous frame
  • On mouse up: redraw with progressively smaller thresholds
  • Consequence: frame rate may vary
Outline

• Motivation
• Point-based rendering
• Point-based surface editing
Pointshop 3D: a surface editing system

- Pointshop 3D
  - Interactive system for point-based surface editing
  - Generalizes 2D photo editing concepts and functionality to 3D point-sampled surfaces
  - Uses 3D surface pixels (surfels) as versatile display and modeling primitive
Pointshop 3D: a surface editing system

- Editing process

Parameterization → Resampling → Editing Operator
Pointshop 3D: a surface editing system

- Editing process
  - Point cloud parameterization \( \Phi \)
    - brings surface and brush into common reference frame
  - Dynamic resampling \( \Psi \)
    - creates one-to-one correspondence of surface and brush samples
  - Editing operator \( \Omega \)
    - combines surface and brush samples

\[
S' = \Omega(\Psi(\Phi(S)), \Psi(B))
\]

modified surface original surface brush

Lin ZHANG, SSE, 2012
Pointshop 3D: a surface editing system

- Parameterization
  - Constrained minimum distortion parameterization of point clouds

\[ u \in [0,1]^2 \implies X(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = x \in P \subset \mathbb{R}^3 \]
Parameterization

Find mapping $X$ that minimizes objective function:

$$C(X) = \sum_{j \in M} (X(p_j) - x_j)^2 + \varepsilon \int_P \gamma(u) du$$
Pointshop 3D: a surface editing system

- Parameterization
  - Measuring distortion

\[ \gamma(u) = \int_{\theta} \left( \frac{\partial^2}{\partial r^2} X_u(\theta, r) \right)^2 d\theta \]

- Integrates squared curvature using local polar re-parameterization

\[ X_u(\theta, r) = X \left( u + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \right) \]
Pointshop 3D: a surface editing system

Overview of the operator framework for point-based surface editing.
Pointshop 3D: a surface editing system

• Editing examples: texture mapping

(a) input texture with feature points; (b) original model with corresponding markers; (c) visualization of the parameter mapping; (d) rendering of the resulting surface
Pointshop 3D: a surface editing system

• Editing examples: Sculpting
  • Apply normal displacements to the surfel positions

\[ x_i' = x_i + d_i \cdot n \]

where \( d_i \) is the displacement coefficient given in the brush function
Pointshop 3D: a surface editing system

- Editing examples: Sculpting
Thanks for your attention